

# Organización del Computador 2

## Segundo parcial – 21/06/18

A  
CORREGIDO  
PUNTO

1 (30)	2 (40)	3 (30)	
25	15	30	70

### Normas generales

- Numere las hojas entregadas. Complete en la primera hoja la cantidad total de hojas entregadas.
- Entregue esta hoja junto al examen, la misma **no** se incluye en la cantidad total de hojas entregadas.
- Está permitido tener los manuales y los apuntes con las listas de instrucciones en el examen. Está prohibido compartir manuales o apuntes entre alumnos durante el examen.
- Cada ejercicio debe realizarse en hojas separadas y numeradas. Debe identificarse cada hoja con nombre, apellido y LU.
- La devolución de los exámenes corregidos es personal. Los pedidos de revisión se realizarán por escrito, antes de retirar el examen corregido del aula.
- Los parciales tienen tres notas: I (Insuficiente): 0 a 59 pts, A- (Aprobado condicional): 60 a 64 pts y A (Aprobado): 65 a 100 pts. No se puede aprobar con A- ambos parciales. Los recuperatorios tienen dos notas: I: 0 a 64 pts y A: 65 a 100 pts.

### Ej. 1. (30 puntos)

Se tiene la siguiente tabla GDT:

Indice	Base	Límite	DB	S	P	L	G	DPL	Tipo
28	0x30100123	0x23F01	1	1	1	0	1	11	0xA
37	0x4A1F0102	0x12222	1	1	1	0	1	10	0x8
39	0x222F22F2	0x40DDD	1	1	1	0	1	00	0x2
43	0x1D900D00	0x5FFFF	1	1	1	0	1	00	0x0

Ver el siguiente esquema de paginación.

Rango Lineal	Rango Físico	Atributos
0x4CA40000 - 0x4CA55FFF	0xD549C000 - 0xD54B1FFF	read only, supervisor
0x55040000 - 0x5504FFFF	0x9A88F000 - 0x9A89EFFF	read/write, supervisor
0x5EDF2000 - 0x5EFF2FFF	0x934F3000 - 0x936F3FFF	read only, user

- (10p) a. Especificar todas las entradas de las estructuras necesarias para construir un esquema de paginación. Suponer que todas las entradas no mencionadas son nulas.
- (20p) b. Resolver las siguientes direcciones, de lógica a lineal y a física. Utilizar las estructuras definidas y suponer que cualquier otra estructura no lo está. Si se produjera un error de protección, indicar cuál error y en qué unidad. Definir EPL en los accesos a datos. El tamaño de todas las operaciones es de 2 bytes.
- I - 00E3:1C9500FF - CPL 11 - lectura
  - II - 012A:0BB50088 - CPL 10 - ejecución
  - III - 013A:3CB00000 - CPL 00 - escritura
  - IV - 015A:3E25FFFF - CPL 00 - lectura

### Ej. 2. (40 puntos)

Considerar un sistema en dos niveles de protección con paginación activa. Este ejecuta concurrentemente 15 tareas una después de la otra.

Una vez ejecutandose, las tareas pueden generar cualquier tipo de excepción, en cuyo caso deben ser reiniciadas. Antes de reiniciar una tarea, se debe informar a todas las otras tareas que una tarea esta siendo reiniciada. Para esto se debe ejecutar la función `void informar(int tarea)` en el contexto

de cada tarea a fin de informar a la misma de dicho suceso. Cada tarea implementa su propia función `informar`, alojando en `0x3700000` la dirección de la función `informar`. Una vez que termina de ejecutar, la función llama al syscall `finalizar` en la interrupción `0x54`.

Para reiniciar una tarea se debe llamar a la función `void limpiarMemoria(int tarea)`, que toma un índice que identifica la tarea y pisa toda la memoria de la tarea por una copia limpia de la misma.



- (10p) a. Describir los campos relevantes de todas las estructuras involucradas en el sistema para administrar segmentación, paginación, tareas, interrupciones y privilegios.
- (30p) b. Escribir en ASM/C la rutina de atención de interrupciones del Reloj, una rutina de atención de excepciones y la syscall `finalizar`.

Nota: Suponer que las funciones `informar` no producen excepciones.

### Ej. 3. (30 puntos)

Se tiene un sistema tipo con segmentación flat y paginación activa que ejecuta concurrentemente una cantidad variable de tareas. Se desea implementar un mecanismo que permita contar, en paginación, a cuantas paginas distintas de usuario se accede entre cambios de contexto por parte de una tarea. El sistema generará un *log* para cada tarea con esta información utilizando la función `void saveCount(int count)` que almacena la cuenta de la tarea actual.

- (10p) a. Idear una solución al mecanismo pedido. Explicar detalladamente su funcionamiento.
- (20p) b. Implementar en ASM/C todas las rutinas necesarias para implementar el mecanismo.

①

# Ejercicio 1.

a) Para construir este esquema de paginación utilizare uno único PAGE DIRECTORY, Primero resuelvo los índices de los directorios binarios

1°

4CA40000  
01001100101001000000  
132 | 240  
PD PT

4CA55FFF  
1010010101010101  
132 | 255  
PD PT

2°

55040000  
01010101010101010101  
154 | 040  
PD PT

5504FFFF  
01010101010101010101  
154 | 04F  
PD PT

3°

5EDF2000  
01011101110111010101  
17B | 1F2  
PD PT

5EFF2FFFF  
01011101110111010101  
17B | 3F2  
PD PT

## PAGE DIRECTORY

TABLA  
PENO  
M1

CR3 >> 12

CR3 + 0x132

CR3 + 0x154

CR3 >> 12 + 0x1FB

PT1 >> 12	ATTRPD1
PT2 >> 12	ATTRPD2
PT3 >> 12	ATTRPD3

Bit P

## 1° PAGE TABLE

PT1 >> 12

PT1 >> 12 + 0x240

PT1 >> 12 + 0x255

0x0549C	ATTRPT1
+ 0x1	ATTRPT1
0x054B1	ATTRPT1

Bit P

## 2° PAGE TABLE

PT2 >> 12

PT2 >> 12 + 0x40

PT2 >> 12 + 0x4F

0x9A88F	ATTRPT2
+ 0x1	ATTRPT2
0x9A89E	ATTRPT2

Bit P

## 3° PAGE TABLE

PT3 >> 12

PT3 >> 12 + 0x1F2

PT3 >> 12 + 0x3F2

0x934F3	ATTRPT3
+ 0x1	ATTRPT3
0x936F3	ATTRPT3

Bit P

(CONTINUA)

Dejé por entendido que yo pongo las posiciones físicas pero pongo las PD y PT1, PT2, PT3, por lo tanto ya están definidas

Luego Defino ATTPD1 = PS=0, A=0, PCBY PWT NO SE ACLARAN (LO MISMO QUE CR3)  
 $\frac{1}{2}S = 0, R/W = 0, P=1$

ATTP1 = G7PAT NO SE ACLARAN, PCBY PWT LO MISMO QUE CR3  
 $\frac{1}{2}S = 0, R/W = 0, P=1, A=0, D=0$   
 (OMITIRÉ G, PAT, PCBY PWT A PARTIR DE AQUÍ)

ATTPD2 = PS=0, A=0,  $\frac{1}{2}S = 0, R/W = 1, P=1$

ATTP2 =  $\frac{1}{2}S = 0, R/W = 1, P=1, A=0, D=0$

ATTPD3 = PS=0, A=0,  $\frac{1}{2}S = 1, R/W = 0, P=1$

ATTP3 =  $\frac{1}{2}S = 1, R/W = 0, P=1, A=0, D=0$

TODO EL RESTO DE ENTRADAS EN PD, PT1, PT2 y PT3 TENDRÁN EL BIT P=0

b. I. 0x00E3: 1C9500FF

$\begin{matrix} 0 & \dots & 011100011 \\ \text{OX } 1C & / & RPL=3 \\ = 28 & & \text{CNT} \end{matrix}$

EL INDICE 28 DE LA GDT ES UN DESCRIPTOR DE SEGMENTO DE TIPO OXA = CÓDIGO EXECUTE/READ y de NIVEL 3

COMO EL BIT DE P=1,  $\max(CPL, RPL) = 3 = DPL$ , VOY A LEER puede utilizar el segmento, fíjate EL LÍMITE y como G=1  $\Rightarrow$

$\text{OX } 1C9500FF + \text{OX } 2 - \text{OX } 1 = \text{OX } 1C9500FF \leq (\text{OX } 23F01 + \text{OX } 1) \ll 12 + \text{OX } FFF$   
 $\text{OX } 23F02FFF$ , ES MAYOR

LUEGO, RESUELVO:  
 $\begin{matrix} & & & & 11 \\ & & & & \text{OX } 30100123 \\ \text{PD1 OX } 132 & & + & \text{OX } 1C9500FF \\ & & \text{OX } 4CA50122 \end{matrix}$  DIRECCIÓN LINEAL

BUSCO EN EL PT1 LA ENTRADA OX250 Y CHEQUEO QUE CUMPLA

ENCUENTRO QUE PD1 Y PT1 ~~12~~ + OX250 TIENEN NIVEL SUPERVISOR ✓  
 Y ESTOY HACIENDO UNA LECTURA A NIVEL USUARIO POR LO TANTO NO ACDO

②

II) 012A:0BBS0088 CPL 10 EJECUCIÓN

$\frac{-0100101010}{RPL=2}$   
0x2 8 CDT

INDICE 3F

GDT [3F] SELECTOR DE SEGMENTO

TIPO 0x8: CÓDIGO SOLO LECTURA.

Y COMO QUIERO EJECUTAR NO PADRE #GPE X SUSTAMENTE  
ES UN SEG. DE  
CÓDIGO

III) 013A:3CB00000 - CPL 00 - ESCRITURA  
RPL=2

GDT [39] SELECTOR DE SEGMENTO TIPO 0x2 DATOS R/W  
- MAX(CPL, RPL) = 2  $\leq$  DPL #GP ERROR ✓

(Como G=1  $\Rightarrow$  CHEQUEO LÍMITE 0x3CB00000 + 0x2 - 0x1 = 0x3CB00001  
 $\leq ((0x40000000 + 1) \ll 12) + 0xFFFF$

0x40000000 POR LO TANTO ES ( $\leq$ )

TODO  
ESTO  
ESTÁ DE  
MAS,  
PENSE QUE  
DPL=2

BUSCO LINEAL

$\frac{0x222F22F2}{+ 0x3CB00001}$   
0x5FDF22F3 DIRECCIÓN LINEAL

PD[0x1FFF] ENCUENTRO QUE EL BIT P = 0

$\Rightarrow$  OCURRE #PAGE FAULT

IV) 015A:3E25FFFF - CPL 0 - LECTURA

RPL=2  
GDT [43] SELECTOR DE SEGMENTO TIPO 0x0 DATOS Read-only  
DPL 0

MAX(CPL, RPL) = 2  $\leq$  DPL = 0

#GP 2 > 0 ✓

③

## 2. a) SEGMENTACIÓN:

Este sistema utilizará segmentación FLAT, tendrá 4 descriptores de segmentos, 2 de código y 2 de datos, cada uno se diferenciará por tener distintas APZ, 2 serán 0 y las otras 3.

Por lo tanto las primeras 5 entradas serán <sup>de la GDT</sup> GDT

0	DESCRIPTOR NULO
1	CÓDIGO NIVEL 0
2	CÓDIGO NIVEL 3
3	DATOS NIVEL 0
4	DATOS NIVEL 3
	⋮

Estos 4 entornos tendrán base en 0x0 y límite de ~~200Mb~~ 100Mb con G=1  
También tendrán R/W=1 y P=1

DESCRIPTORES TSSs

## PAGINACIÓN:

Cada tarea tendrá su propio directorio de páginas los cuales mapearán las mismas direcciones virtuales pero a diferentes direcciones físicas, excepto las 10 Mb de Kernel que serán identity MAPPING.

PA		P
0	DIRPT1	ATT-K
1	DIRPT2	ATT-K
2	DIRPT3	ATT-K
	⋮	0
	⋮	⋮
	⋮	0
13	DIR_PT13	ATT-T
	⋮	1
	⋮	⋮
25	DIR_PT25	ATT-T
	⋮	0
	⋮	⋮

Cada cada PT puede direccionar 4Mb de memoria, utilizo 2 tablas completas y la tercera por lo mismo. ATT-K serán los atributos necesarios para acceder al kernel, como  $U/S=0$ , LOS MISMOS UTILIZADOS POR TAREAS. Mientras que los atributos de las PT1, PT2 y PT3 serán iguales entre si, como  $U/S=1$  R/W=1, ATT-T PT33 solo utilizo la mitad de la tabla para abajo

## TAREAS:

Como son 15 las tareas que se ejecutan tendré 15 TSS más uno extra para la tarea inicial, estos TSS tendrán sus ~~inici~~ descriptors de TSS alojados en lo <sup>GDT</sup> TSS, en los índices 14 a 25 ~~ordenados~~ numéricamente, en la 10 estará la tarea inicial, Asumo que los TSS tienen ~~una fase particular~~ cada uno su información particular. Entre ellas tendrán en CR3 lo propio.

En SS el índice 4 <sup>de GDT</sup>, también lo tendrán GS, FS, DS y ES.

En CS estará el índice 2 <sup>GDT</sup>. El EIP estará apuntando al inicio de la tarea.

El ESP y EBP apuntarán dentro del ~~lo~~ espacio de la tarea. EFL1565 = 0x202   
 para tener interrupciones.

En SS0 estará el índice 3 de lo GDT y el ESP0 apuntará a su pila de Kernel.

## Interrupciones:

Se tendrá uno IDT con las primeras 32 entradas mapeadas a la misma resolución de excepción. Todas estas son interrupciones de Hardware por lo tanto tendrán DPL=0.

También estará presente el índice 32 en lo IDT que se refiere a la interrupción de clock lo cual también tendrá DPL=0.

Finalmente habrá uno índice más presente, el 0x54 que tendrá la interrupción finalizar y tendrá DPL=3.

# SCHEDULER:

Tendré varias estructuras, las cuales utilizaré en distintos momentos:

- Uno array de 15 <sup>15 que</sup> donde estarán las TSS iniciales de cada tarea ordenadamente. Se llamará TSS\* TSS\_iniciales
- ~~TSS\*~~ ULTIMO-~~TSS~~ <sup>TSS[15]</sup> DONDE GUARDARE el ~~de~~ <sup>estado</sup> ~~TSS~~ antes de ser enviado a informar.
- INT HAY\_TAREA\_REINICIANDOSE que estará en 1 cuando se este reiniciando una tarea. y 0 en el caso contrario.
- INT TAREA\_REINICIANDOSE que tendrá el índice identificador de tareas

b)

.Text

- ISR-32:

.DATA

OFFSET: DD 0x0

SELECTOR: DD 0x0

PUSHAD

CALL FIN-INTR-PICT

CALL PROX-TAREA

MOV [SELECTOR], AX

CALL GET-GDT

~~MOV~~ PUSH EAX

MOV EAX, ~~PROX-TAREA~~ TR

PUSH EAX; ~~MOV EAX, SEGUIR~~ PUSH EAX

CALL INFORMAR-SI-CS-NECESARIO

~~seguir~~ JMP FAR [OFFSET]

POPAD

IRET

Ⓢ & ! (TAREA == TAREA -  
REINICIANDOSE)

VOID INFORMAR-SI-CS-NECESARIO (GDT\* DIR-GDT, INT TAREA) {

Ⓢ A LA DEBIDA

IF (HAY-TAREA-REINICIANDOSE == 1) { Ⓢ IF (TAREA == 15) TAREA = 0; } // CASO TAREA 15.

ULTIMO-TSS[TAREA] = \*(DIR-GDT[TAREA+1].dir-base); // ME MUEVO A LA PROX TAREA

TSS\* TSS-ACTUAL = DIR-GDT[TAREA+1].dir-base;

TSS-ACTUAL.eip = 0x3f000000

Ⓢ TSS-ACTUAL.eip = TAREA-REINICIANDOSE;

} Ⓢ ELSE IF (TAREA == TAREA-REINICIANDOSE) { LIMPIAR MEMORIA (TAREA) } }

RESTAURAR-TAREAS (TAREA)

HAY-TAREA-REINICIANDOSE = 0

?  
X  
MAL

ISR.N:

~~POPAD~~ PUSHAD

~~CALL PREP~~

MOV EAX, TR

PUSH EAX

CALL PREPARAR-Reinicio

~~POPAD~~

~~JMP FAR INLC~~ JMP FAR INLC

~~IRET~~

POPAD

IRET

INLC = TAREA

SIMILAR A TP, ESPERA  
AL CLOCK SIN MODIFICARSE

VOID PREPARAR-Reinicio (INT TAREA) {

HAY-TAREA-Reiniciandose = 1;

TAREA-Reiniciandose = TAREA;

ULTIMO-TSS[TAREA] = TSS-INICIAL[TAREA];

}

Se PISA EN EL  
JMP FAR

Finalizar:

~~PUSHAD~~

~~MOV EAX, TR~~

~~PUSH EAX~~

~~CALL RESTAURAR-TAREA~~

~~JMP FAR INLC~~ JMP FAR INLC

POPAD

IRET

~~RESTAURAR-TAREA (TAREA) {~~

~~TAREA-RESTAURADA[TAREA] = 1;~~

ME FALTA RESTAURAR LAS TAREAS LUEGO DE

FINALIZAR, MI IDEA ERA

VOID RESTAURAR-TAREAS (~~GDT~~) {

FOR (INT i=0; i<15; i++) {

IF (TAREA-Reiniciandose ~~se~~  $\neq$  i) {

\*GDT[i+10].dir base = ULTIMO-TSS[i];

}

MAL

PERDON POR DESPREZOS

⑤

3. PARA ESTE EJERCICIO SE ~~SE SUPONDRÁ~~ SUPONDRÁ QUE LAS TAREAS QUE SE CORREN SON NIVEL 3, DE ESTA MANERA ME ASEGURO QUE NO PODRAN ACCEDER A LAS PD, PT PARA MODIFICARLAS, TAMBIÉN SUPONGO QUE EL ENCARGADO DEL CAMBIO DE TAREAS ES LA EXCEPCIÓN DE CLOCK <sup>④</sup> (ISR-32) <sup>④</sup> (TODO ESTO FUE CONSULTADO)

POR LO TANTO SE TENDRÁ UNA FUNCIÓN <sup>④</sup> QUE SERÁ LLAMADA DURANTE LA EXCEPCIÓN DEZ CLOCK. ESTA FUNCIÓN RECIBIRÁ COMO PARÁMETRO LA DIRECCIÓN DEL DIRECTORIO DE TABLAS DE PÁGINAS. CON ESTE LOOPEARA DENTRO DE LAS PAGE TABLES, SIEMPRE QUE ~~EN~~ TENGAN EL BIT P EN 1, Y CONTARÁ LA CANTIDAD DE ~~TAREAS~~ PÁGINAS CON EL BIT ACCESSED EN 1 Y U-S EN 1, MIENTRAS AL MISMO TIEMPO PONE ACCESSED EN 0. UNA VEZ FINA LIZADO EL LOOP SE LLAMARA A SAVE COUNT (COUNT) CON EL NUMERO OBTENIDO. FINALMENTE VOLVERA A LA EXCEPCIÓN DE CLOCK Y ESTA ~~HARA~~ HARA EL SWITCH DE TAREAS.

④ LA FUNCIÓN SERA = VOID CONTAR (-PD\* DIR-PD)

④ ADemás SE SUPONDRÁ QUE AL DECIR QUE ES UN SISTEMA TIPO <sup>EN EL ENONC</sup> TENDRÁ EL KERNEL Mapeado con identity mapping y LA PT y PD ESTARÁN DENTRO. TAMBIÉN SE TENDRAN LAS ESTRUCTURAS PT Y PD IMPLEMENTADAS EN C.

ASUMO QUE PÁGINA DISTINTA SE REFIERE A DISTINTO DESCRIPTOR DE PÁGINA Y DISTINTO DESCRIPTOR DE DIRECTORIO.

b) .TEXT:  
• ISR-32:

.DATA

OFFSET: DD 0x0  
selector: DW 0x0

```
PUSHAD  
CALL FIN_INTR_PIC1  
MOV EAX, CR3  
PUSH EAX  
CALL CONTAR ✓  
CALL PROX-TAREA  
MOV [selector], AX  
JMP FAR [OFFSET]  
POPAD  
IRET
```

```
VOID CONTAR ( - PD.. * DIR_PD) {  
    INT PAGES = 0;  
    FOR ( INT i = 0; i < 1024; i++ ) {  
        IF ( DIR_PD[i].P == 1 && DIR_PD[i].U.S == 1 && DIR_PD[i].A == 1 ) {  
            PT * DIR_PT = DIR_PD[i].base_PD < 12;  
            FOR ( INT j = 0; j < 1024; j++ ) {  
                IF ( DIR_PT[j].P == 1 && DIR_PT[j].U.S == 1 && DIR_PT[j].A == 1 ) {  
                    PAGES++;  
                    DIR_PT[j].A = 0; ✓  
                }  
            }  
            DIR_PD[i].A = 0; ✓  
        }  
    }  
    SAVE COUNT ( PAGES ); ✓  
}
```